ZZJ's Barn Door Tracker

< Design | Operation | Math | Electronics | Software | Materials | My Build | Advice / Where I messed up | Links >

First Light! M42, The Orion Nebula

I've always liked astrophotography, but it always seemed so expensive to get into, beyond short wide angle exposures. Up until July of last year, I didn't even have a suitable camera for astrophotography, but with the Canon EOS M fire sale, suddenly it was a possibility, as the EOS M has a nice large APS-C class sensor in it. But, without a tracking mount, there's no way to do an exposure longer than 15 seconds or thereabouts (@35mm). There are nice ready made camera tracking mounts, but they go for $400 - by themselves. I'd have to buy a heavy-duty tripod, a ball mount, and an alignment scope, too. Or, buy a telescope mount with a guide motor, which puts you back roughly as much. Astrophotography is cool, but it's not quite $600+ cool at this point in time. Plus, *if* I had to spend the cash, I'd rather spend it on a better telescope or camera lenses, not an extremely specialized camera mount. So, that shelved my dreams until recently.

Once, I had run across the idea of a barn door tracker, which seemed like a clever but more importantly a *cheap* solution to longer exposures. However, the basic design can only do short exposures, and the dual arm tracker invented by Dave Trott required a lot of relatively fine fabrication that I wasn't too sure if I could pull off. In light of that, I forgot about them for a while. Then, I saw the Cloudbait observatory tracker build, which introduces a more elegant solution: controlling error by varying drive speed, rather than mechanically compensating like the dual arm design.

I tried and failed to make sense of the cloudbait excel spreadsheet, which he used to calculate his step rates, but decided to try building my own anyways. I looked through my collection of electronic bits 'n bobs, and found a stepper and its worm drive in an old HP printer carcass; a serious shortcut. Ordered a camera ball mount and A4988 stepper driver, made a pivot for the salvaged stepper, made a pivot nut, then bought some wood, blah, blah, work, math, etc and then I had this tracker dingus.

It's an isosceles-type barn door tracker (pivot on motor and drive nut) with AltAzimuth adjustments, as it was to be mounted on a stand without any adjustments rather than an expensive tripod. An ATTiny85 is the brains of the operation, running off a precalculated table for step rate vs. time. I've been able to make two minute exposures at 250mm with room to go longer. I've been having some issues earlier in the run at that focal length, but I think I've gotten those fixed, pending testing (clouds). Shorter focal lengths have been bang on over 5 minutes at any point, though.

- 70 minutes of runtime
- -12 to +42 degrees azimuth adjustment
- +21 to +47 degrees altitude adjustment
- Approx 5 days of runtime on 7ah battery.
- *Theoretical* error of less than 0.005 arc second (chart)

Design

Mechanical design is nothing fancy - the tracker is just two hinged boards being pushed apart by a threaded rod, driven by a stepper motor. A finderscope is mounted near the hinge, and a ball mount is plopped in the middle of it (**the location of the ball mount does not particularly matter - took me forever to figure that out!**). Another piece of threaded rod controls the altitude of the tracker and the final bit of rod is used to

rotate the base of the mount (azimuth) - my workaround for not wanting to buy a proper tripod. The microcontroller is what makes this mount special, and lets it actually track for any extended period of time.

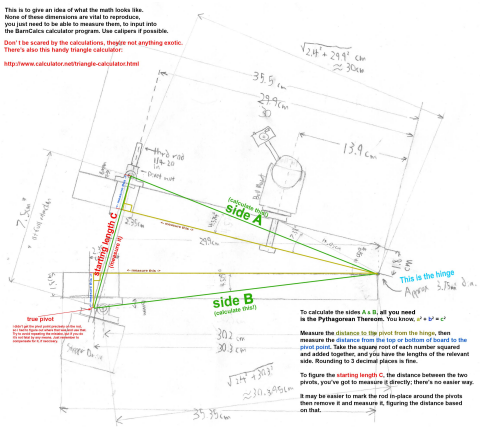
Considerations for building your own:

Polaris's altitude is roughly equivalent to your latitude.

- The precision of fabricating the tracker is less important than your ability to measure what you've done.
- If you are going to build axes, you only don't really need all that much travel - they're for fine alignment. That I ended up with so much adjustment was just a happy accident. Keep in mind the altitude adjustment needs to cover the range where polaris normally is, though. Down in Texas, that's about 30 degrees. **See the map to the right - Polaris's altitude is roughly equivalent to your latitude.**
- Remember that everything I'm talking about is in the context of me being able to control the motor rate... if you're building a manual mount, this guide/write up probably won't be of much use to you. Even if you are planning to go with a powered mount, I'd still consider something like the curved rod tracker described here.
- The precision required for measurement is directly proportional to what focal lengths you want to use. For relatively short focal lengths, you can easily get away with lazy alignment, build/measurement errors, and so on. For short exposures you can even get away without changing the RPM at which the shaft is turned, for a short time. But if you want to use longer focal lengths, precision is key.
- With this mechanical approach, tweaking of code **will** be required - this isn't exactly a grab and go solution, and I'm not sure if it was the best route as I'm still having some tracking issues at the longest focal lengths of my lens. The curved rod trackers deserve serious consideration. Though, it seems like most of the time people use shorter lenses on barn door trackers...
- If you're going to directly drive the rod with a stepper, I'd recommend using micro stepping in order to produce smooth movement, and make sure the stepper has relatively small steps to begin with. An alternative is gearing it down, which isn't necessarily a bad idea anyways.
- Use particleboard or plywood, if possible, not planks of wood like I did. I selected one that was fairly flat and it's still not perfect. Not sure if it's giving me trouble or not but I'd just have soon not had to worry about that in the first place.

Math

Generally, RPM must decrease over time to maintain the same angular velocity (chart), in an isosceles tracker. The math honestly is not too scary - it's just the law of cosines. The derivative of the law of cosines is used to calculate the initial rate, but after that it's just a bit of brute force. Note where graph shows the RPM oscillating between two values - this is where an integral value would be too slow or too fast to match the ideal tracking rate. So, by toggling between rates, it achieves a correct average rate.



Here's a diagram of the math involved, and how to get the input values for the calculator program.

Side *A* is the distance from the pivot of the drive screw to the hinge. **Side *B*** is the distance from the pivot of the stepper to the hinge. The **starting length *C***, is the length of the drive rod from the center of each pivot. These are the factors which must be measured for the calculator.

To figure out the lengths of sides ***A*** and ***B***, measure the distance from the board to the pivot point for both top and bottom, then measure the length along the board to the hinge. Use the pythagorean theorem to calculate the side length. Alternatively, use this nifty triangle calculator page, which can handle the math for you. You could try directly measuring, but that's a lot harder to do. You'll also have to directly measure the length of the drive rod between the two pivots, which is the **starting length *C***.

Formulas

These are the two formulas the calculator uses in figuring out the table of values used.

This is the Law of Cosines SSS case, which evaluates to the angle ***W***, opposite side ***C***. Triangle sides ***A***, ***B***, and ***C*** are the known input values. This is used to by the calculator to find angle ***S***, the initial angle between the two pivots.

$$W = \cos^{-1}\left(\frac{A^2 + B^2 - C^2}{2\,A\,B}\right)$$

This is a derivative of the Law of Cosines SAS case. This gives the rate of change at time ***t***, with ***V*** being the desired tracking rate, ***S*** the starting angle, and ***A*** and ***B*** the two sides of the triangle. Note, this is not a general derivative, as it uses the function for determining target angle at time ***t***; **W(*t*)** = ***S*** + ***t*** * ***V***.

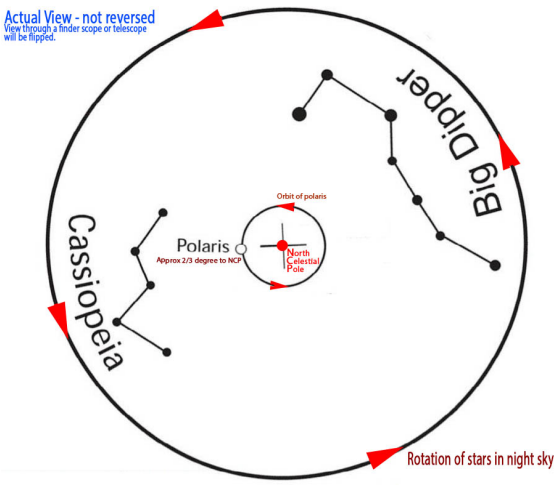
$$C = \frac{A\,B\,V\sin(t\,V + S)}{\sqrt{-2\,A\,B\cos(t\,V + S) + A^2 + B^2}}$$

Main code (zip file) of the calculator is pretty simple: the program selects the most accurate delay at ***t***=0, using the derivative above, then every $INTERVAL steps it calculates the error at the current rate, and if it reduced by slowing down or speeding up. If a rate change is beneficial, then that is noted, and so on. Repeats until $MAX_TIME is reached, and then prints out the resulting table of values noting where corrections to the rate should be made.

Operation

On power on, it will rewind at 7.1x rate until the limit switch is hit, at which point it advances slightly then waits for the user to press the button, at which point it will run forward at the sidereal rate. If the button is pressed again, it will rewind again, and so on. Button presses while rewinding do nothing except pause it while button is held. Pretty much foolproof.

To actually use the mount, you have to align it so the hinge points to Polaris, The North Star. First, you point the tracker roughly north, and set the altitude to what is correct for your latitude - Polaris's altitude is roughly equivalent to your latitude. After that, manipulate the axes so that polaris is centered in the finder scope (if present) and/or the hinge is pointing at polaris. Depending on the focal length of the lens you intend to use, this may be all that is required.


Actual View - not reversed
View through a finder scope or telescope will be flipped.
Big Dipper
Cassiopeia
Polaris
Approx 2/3 degree to NCP
Orbit of polaris
North Celestial Pole
Rotation of stars in night sky
A diagram of how the stars rotate in the night sky, relative to polaris and the north celestial pole.

If you intend to use a longer focal length, you will have to align to the **north celestial pole**, the true point at which stars rotate about; it lies about 2/3s of a degree away from Polaris. First, align the mount to Polaris. Then, look for either cassopeia or the big dipper, depending on time of year, to determine the angle the NCP is relative to polaris. Here's a diagram to help with that, and here's the reversed version; how it looks through a finder scope. I printed the reversed diagram, "laminated" it with some tape and cardboard, then stuck a pin in the center on the top of the tracker, so I can rotate it relative to the position of the constellations. You could also use Stellarium to figure it out, too - I find turning on both equatorial and azimuthal grids make it pretty easy to judge the angle, in practice, I use that more often than the disk. You can even get an android version of stellarium.

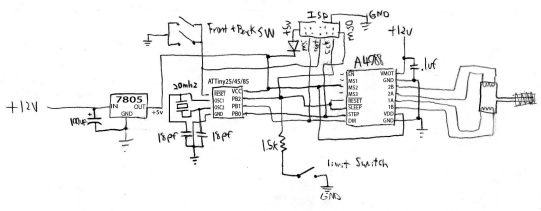
Once you have determined the angle to the NCP, move the axes approximately 2/3 of a degree at that angle. Through the finderscope, I can see HIP 7283, a star near polaris, so I use the distance between it and polaris - **1/3** of a degree - to figure the distance I need to move. Once you've moved to where the NCP should be, you

are now roughly aligned. There will be alignment error between the scope and mount, the angle won't be perfect, and the distance won't be exact, but to what degree this effects you depends on the lens you are using. You may need to take exposures and make tweaks to the axes until no trailing is evident, if you are using an long telephoto lens or have a telescope mounted. I think anything but the shortest focal length telescopes are probably beyond this type of mount, though - would be too hard to align and required precision of tracking too high.

I also made an remote/intervalometer out of another ATTiny85 to operate my camera, which you can see in the first picture. It should work for any Canon camera that works with the Canon RC type remotes. Remote source and circuit diagram. With an IR led stuck on the ball mount, it's just like a wired connection. Once I have the tracker set up and camera aimed, I can walk away while the camera happily takes exposures without me. It runs on 3 AA batteries, which should last for a very long time. It could last even longer if I had the AVR sleep between pulses.

Electronics

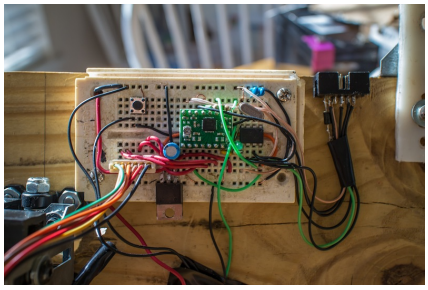
I used a A4988 stepper driver from Pololu, at 1/4 microstepping (selected for max smoothness). Stepper motor is a 7.5 degrees/step bipolar motor sourced from an old HP printer. After the worm drive reduction (55:1) and microstepping (1/4), we're looking at 10560 steps per full revolution. A 7805 is used to provide logic power.



my handwriting is atrocious...

For the AVR, I used a Attiny85 as it was what I had handy, but you could get away with a Attiny25 just fine. My code, with a 548 byte table for 1 hour of running, takes a bit under 1kb of flash. You could just easily use a bigger uC (more pins) such as an ATtiny84 or ATMega part, I just didn't have space left on the beater breadboard that I didn't mind semi-permanently installing. Whatever you use for the controller, if you want to use longer focal lengths, you will need a crystal, as the RC oscillator built into an AVR is temperature sensitive and inconsistent. Arduinos and most other dev boards already have a 16mhz crystal onboard.

Circuit description: AVR Pin 5 is connected to the step input of the A4988, pin 6 is connected to the direction input of the A4988, and pin 7 is connected to the SLEEP (6) and RESET (5) lines of the A4988, as well as the limit switch via a 1.5k resistor. The crystal and its caps occupy pins 2 and 3. The AVR's pin 1 (RESET) is connected to the pushbuttons for user input (which connect to ground). A 100uf electrolytic capacitor and 0.1uf ceramic capacitor are connected between +12v and ground. A4988 pin 3 (MS2) is pulled high to enable 1/4 stepping mode. An ISP header is present and connected to the relevant pins as well as hooked to the +5v bus via a diode to allow programming without the battery attached.



The header to the right is the ISP header, to allow me to program the microcontroller without removing it.

Circuit uses about 50-60ma while running or rewinding, and about 4ma idle. It could run for something like 5 days straight, too bad it'd run out of rod after 70 minutes!

The intervalometer/remote is not much more than a transistor, 2 IR LEDs, attiny, and a few switches. See the remote circuit diagram for details. I also got a third IR LED, put salvaged connector on it, and glued that to the camera part of the ball mount. This allows me to control the EOS M via a wired connection rather than having to reach around with the remote or place the remote where it can be seen by the camera's IR receiver. Plus, once I get it aligned and set on a target, I can leave it alone while it takes pictures for me!

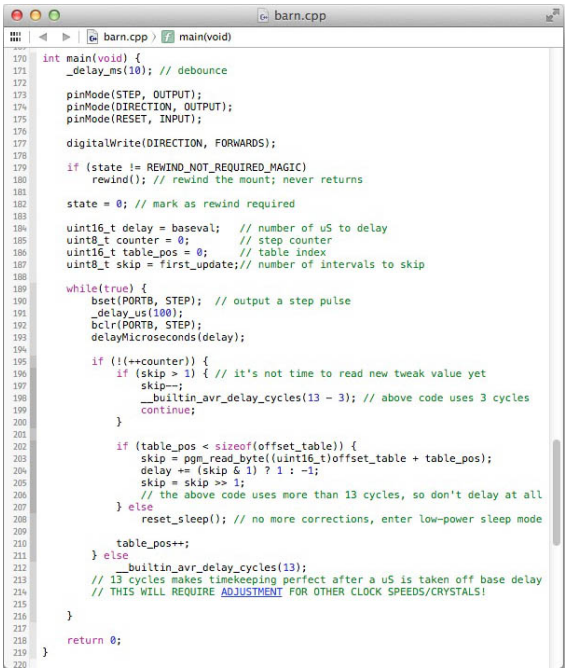
I also ended up making a power adapter for my camera, as it is something of a battery hog. Why bother with little tiny batteries when I have this nice big one? The canon solution is a dummy battery with a power jack in the bottom; the EOS M has a rubber flap on the bottom of the battery door to allow access to the jack. I made a dummy battery out of a solid chunk of milkjug-derived plastic, added contacts and a jack, salvaged a LM317 adjustable regulator (set it to 7.7v), and stuck that in an AC adapter's casing; done! It looks nice

plugged in, too. It's only about 60% efficient, as the LM317 burns waste energy as heat, but it's still enough for over 6 hours of use with a 7AH battery with the tracker active.

Software

BarnCalcs is the C program I wrote to calculate the delay table. (zip file) It should be pretty easily adapted to any other isosceles type barndoor tracker, though you will likely need to modify it to use milliseconds per step rather than microseconds per step. I've done a bit of work towards that end, it should not be difficult to finish the job. Theoretical accuracy is under 0.005 of an arcsecond. In reality, it depends on your measurements and how well you've aligned the mount, as well as how accurate the drive rod is. Click here for an overview of how the program works.

The AVR code (zip file) is simple: it either rewinds or runs forward. When running forward, it steps; delaying a specific amount between each. every 256 steps it will either mark an interval off, or update delay value and load the number of intervals til next tweak. At the end of the table, it stops. Not much more to it. When rewinding, it goes until it hits limit switch, advances a little, then fully shuts down the AVR after putting the A4988 in a sleep state. It uses very little power like this; the circuit could last for weeks in standby mode.


barn.cpp
barn.cpp
main(void)
int main(void) {
_delay_ms(10); // debounce
pinMode(STEP, OUTPUT);
pinMode(DIRECTION, OUTPUT);
pinMode(RESET, INPUT);
digitalWrite(DIRECTION, FORWARDS);
if (state != REWIND_NOT_REQUIRED_MAGIC)
rewind(); // rewind the mount; never returns
state = 0; // mark as rewind required
uint16_t delay = baseval; // number of uS to delay
uint8_t counter = 0; // step counter
uint16_t table_pos = 0; // table index
uint8_t skip = first_update;// number of intervals to skip
while(true) {
bset(PORTB, STEP); // output a step pulse
_delay_us(100);
bclr(PORTB, STEP);
delayMicroseconds(delay);
if (!(++counter)) {
if (skip > 1) { // it's not time to read new tweak value yet
skip--;
__builtin_avr_delay_cycles(13 - 3); // above code uses 3 cycles
continue;
}
if (table_pos < sizeof(offset_table)) {
skip = pgm_read_byte((uint16_t)offset_table + table_pos);
delay += (skip & 1) ? 1 : -1;
skip = skip >> 1;
// the above code uses more than 13 cycles, so don't delay at all
} else
reset_sleep(); // no more corrections, enter low-power sleep mode
table_pos++;
} else
__builtin_avr_delay_cycles(13);
// 13 cycles makes timekeeping perfect after a uS is taken off base delay
// THIS WILL REQUIRE ADJUSTMENT FOR OTHER CLOCK SPEEDS/CRYSTALS!
}
return 0;
}

Here's a commented AVR ASM listing of the main loop.

In the AVR code, I used a clever trick to allow the RESET pin to be used as the user input. On start, a variable specially marked not to be initialized is read; if it's not a special value, a rewind is executed, and at the end of the rewind, the special value is stored in the variable. If the user resets the uC via the buttons connected to RESET, the memory will still contain the special value, so when the program starts again the code will read the var, see the special value, and then run normally after clearing the variable. This allows me to not have to burn the "reset as IO" fuse bit, which would prevent me from programming the AVR without removing it. Turns out to have been a godsend as my initial delay table wasn't even in the right ballpark. Of course, this is a non-issue if you use a chip with more pins.

My code uses a delayMicroseconds for a 20mhz uC; you will need to use a different one if your crystal value differs. Paul Stoffregen's code will work; it can deal with clocks of 1mhz, 2mhz, 4 mhz, 8mhz, and 16mhz. I used it as a base for my 20mhz routine.

I used a chinese Saleae Logic clone to figure out the number of dead cycles to add to get timing accuracy just right - ended up about 400 microseconds fast over 70 minutes; plenty accurate. Once timing is right and you have the drive characteristics figured out (ie. steps/rev, threads/inch, etc), the only place for error is in the measured values you put into the calculator program. If you are using a lower gear ratio or direct drive and/or intend to only use wide lenses, you could skip calibrating the delay entirely.

The remote/intervalometer code (zip file) is extremely simple. At power up, it waits 10 milliseconds for button debouncing, sets the delay value based on the state of the switch, and sends out a "take picture" IR sequence - 16 pulses @ ~32khz, 7.6ms delay, 16 more pulses. After this point, if the user is using the momentary push button, they presumably let go and power is lost. If they used the power SPST switch, then 60-300 seconds later the camera sends a second IR sequence to terminate the bulb exposure, waits 2.5 seconds for the camera to recover its wits, then loops, sending out another IR sequence, etc...

Programming the AVR was done via ArduinoISP running on a Teensy 2.0 until I killed it in the same incident that claimed the first stepper board (12v to logic supply), I then programmed it via a ATTiny-based usbtinyisp programmer.

All development was done on a Mac, using the avr-gcc toolchain and Xcode as an editor. These programs should compile fine under any *nix based OS, windows would need to edit the makefiles. No libraries are required; each program is standalone.

Materials

Tools

- Drill press & bits
- Dremel w/ cutting disks
- Heat gun (to fuse together & bend plastic)
- Tin snips (to cut mecanno strips)
- Hacksaw
- Calipers (important)
- Screwdrivers
- JB weld, Superglue
- Soldering iron & accessories

Materials I had

- ATTiny85 microcontroller (1.3$/per)
- Stepper motor and worm drive (src: old HP printer)
- Some mecanno-type metal pieces and machine screws
- Meade ETX-70 tripod
- Beat up radioshack breadboard
- 20mhz crystal & caps
- 7805 Regulator
- Penny
- Big ass azimuth axis pivot bolt and washers
- 3/8in ball mount bolt
- Misc hardware (mostly small wood screws, and some machine screws)
- Compass
- Brick
- Small hinge
- Milk jugs
- Remote components (attiny, transistor, IR leds, switches)

Stuff I had to buy

- 58$ 9x50mm Orion finderscope
  This scope is way overkill; you would be fine with this 6x30 which mounts in the same manner. There are also specialized polar alignment scopes. If you aren't planning to use longer lenses, you can even omit a finder entirely.
- 12$ Polou A4988 Stepper driver
- 17$ Ball Mount
- 10$ 7ah 12v Lead acid battery
- 3$ 3ft 1/4in-20 threaded rod
- 5$ 1x6in wood, 6 ft
- 10$ misc hardware - wood screws etc
- 8$ 1ft Piano hinge

Wastage

- 12$ stepper driver and 16$ teensy (killed by 12v)
- 10$ airsoft scope (unsuited for purpose)
- 4$ worth of PVC fittings (failed ball mount)

My Build

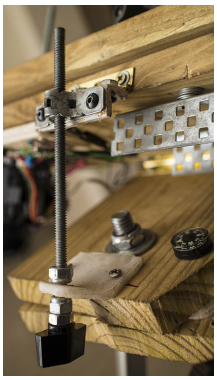
Basic construction is easy. Cut two boards off the plank - I used 14in, but it doesn't particularly matter. Drill a large hole (or cut a slot) in one end of the 14 inch boards for the drive rod, put half of the piano hinge at the other side. Mount the motor and drive nut however you see fit. If you want axes, cut two more board off the plank - I used 7 inches. Drill a nice big hole through each of the 7 inch boards, put a bolt through them with washers on top, bottom, and in-between boards. Cinch it up fairly tight so there is resistance to turning. Put a piano hinge at one side of the upper 7in board, and screw one side to the bottom of the tracker. You could get away without making a screw driven azimuth adjustment, but you'll definitely need something to control the altitude (elevation).

The brick and battery improve stability of the tripod and damp vibration.

Finderscope bracket: Mounted by drilling a single hole in it and running a screw through it. A receiver for the metal tab thingie was dremeled out of the metal it's mounted on. A more elegant solution would be to purchase a proper dovetail receiver, but this works just as well and I had the parts handy. I sighted in the scope by making a reflex sight out of a straw and some thread, then taped to the hinge and sighted it in during the day.

Limit switch: Wire soldered to penny and penny superglued to base. A small bolt was run through the top board and a wire connected to it, altogether forming the limit switch used for rewinding. One side connected to ground, other to pin 7 of the attiny85 via a 1.5k resistor. A microswitch would work just as well, if you have one handy. Probably a better idea, honestly...

Altitude Axis



Top mount was created out of plastic bag material rather than milk jugs. Recommend milk jugs, plastic bags are a lot harder to melt down into a nice chunk of plastic stock. It's the same kind of plastic, though. Heated it up with the hot air gun after fusing bags together, then bent it at a 90 degree angle with the aid of a piece of wood. Cooled it again and drilled the holes in it. Two nuts are held by the metal strap against the plastic.

The bottom part, as seen in this image, is where the milk jug came in. Fused together a nice slab of plastic from milk jugs, bent it slightly, and drilled the relevant holes in it. A single nylock nut rides against it.

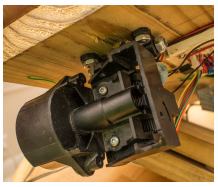
An earlier version of the altitude adjustment is where I figured out how to make plastic chunks out of milk jugs. Over a few design iterations, I had found that a slippery plastic washer gave a nice smooth adjustment against the original metal mounting, rather than nasty, gritty metal on metal contact. However, there was some flex in the bodged up metal support, and the plastic washer was fiddly - it was possible to knock the mount out of alignment by lifting the board up slightly and having the washer settle in a different manner. So, I tried to find a chunk of a similar plastic (HDPE) that I could make a new, stronger mount out of. Failed. Despaired. Decided to try melting milk jug fragments together. Eventually succeeded with the help of a hot air gun, and made a nice 1/4in thick slab of plastic. Cut it down, bent it, drilled a few holes in it and stuck it on. Minimal flex and silky smooth movement. Very strong, too.

Azimuth axis: Two board chunks are mounted with one big bolt with a washer between the boards and a nylock nut. The bolt is cinched up enough to provide minor *resistance* to turning - but not too much. Receiver of the rod (near the hinge) was made out of another chunk of milk jug derived plastic, bent with a hot air gun. The blue things and the washers in the middle of the receiver are so it can't flex. For the pivot, I went through about 5 different designs trying to reduce slop in the axis, but this final overbuilt design has reduced slop to nearly nothing.

Azimuth axis pivot nut: This was a normal nut that I ground down with a dermal and some cutting disks to have two lugs which fit nice and tight into the holes of mecanno.

Mecanno/erector set metal strips are sold by lowes as furniture repair strips. Note, the stuff lowes' sells is really brittle but strong; if you are going to bend it, don't bend it sharply and suddenly, it will just snap. Do it slowly and gently, heating it might help. It's pretty cheap, and cuts fine with tin snips, or even medical shears.

Mount drive pivot nut: tacked 3 nuts together with solder, then stuck two machine screws on the middle nut with superglue. Slathered the lot with JB weld. Two vex parts are used for the pivot, but you could use carefully bent mecanno strips too, or make some out of milk jug plastic.



Servo drive: In a HP printer carcass, a stepper was set up with a worm drive already to raise and lower the print head. Just what I needed. I removed the assembly from printer, cut off all the extra bits of plastic, opened one side of the gear bit, screwed the threaded rod into the gear, and mounted it on a pivot made out of some plastic bits from a VEX set. Very easy. Stepper is a bipolar/unipolar type, don't know more about it aside from stepping 7.5 degrees per step and a coil resistance of approximately 78 ohms. Worm drive is a 55:1 reduction. Overall, a serious score; possibly wouldn't have even made this mount if not for finding it. Here's another view.

Meade tripod: I was dreading the idea of having to buy a real (read: expensive) tripod, then realized I had the tripod for my old ETX-70 telescope. It's not ideal, but it's fairly sturdy, especially with the addition of a 8lb brick (and 4lb battery) hanging off some paracord underneath it. The telescope mounted by two threaded inserts in its base; I drilled two large holes at the appropriate spots in the bottom bit of wood, forced a nut in there with superglue, and superglued a large washer over each. 3 furniture feet were stuck on to the base of the mount; done. Prior to this, I was planning on making a stand out of some PVC, something like this.

Ball mount: Seriously, just buy one, like this. It's extremely solid, has a quick release plate, and is a steal of a deal. I tried to improvise one out of some PVC couplings and a wood ball, but even if it had worked the result would have been extremely crude to use, especially without knocking the mount out of alignment, and severely limited as to max altitude. Whatever you do, **the location at which the ball mount is mounted doesn't matter**. Took forever for me to figure that out. Well, it should be on the top board. It'd be bad if it wasn't. But other than that, it's what's convenient for you. I centered mine on the point where the altitude rod attached, to reduce flex.

More Build Advice / Where I messed up

If you are going to use a finderscope, don't try to skimp and use a cheap rifle scope. The eye relief on those things is murderous - the required distance from scope to eye, and the image is rather dim due to the small aperture. So basically, I couldn't tell when my eye was in the right place or not because all I saw was black. It might be possible to rig something up (maybe some PVC?) to act as a guide... but I wouldn't bother unless you have a free rifle scope already.

An alternative to a finder scope is a reflex sight (no magnification), such as a straw with a crosshair or telrad finder, but be advised that a reflex sight will be just about useless in an area with light pollution, or if you have shitty eyes (like me).

Initially I had 2+ turns of dead space in the azimuth axis, but I've found that alignment is crucial at longer focal lengths; sometimes I've had to fine tune alignment by making exposures and tweaking the axes until there is no visible trailing. The dead space was murderous when trying to keep track of how many turns I tweaked, so ended up reinforcing it a lot to get down to nearly no slop. Not bad for something bodged up...

Watch out for the motor power... I fried my poor teensy microcontroller and a stepper driver by accidentally connecting motor power (+12v) to the logic power bus (+5v). You can get lower voltage stepper drivers such as the DRV8834 (the A4988 requires 7-35v).

You can get useful bits of plastic stock by cutting up a milk jug into strips then melting them together with a heat gun. Put a strip down, heat it, put another on top, heat, etc. Apply pressure with something flat like a piece of wood every few strips. The resulting material is very easy to drill, easy to cut with a hacksaw, easy to dremel, pretty much a dream to work with. Very strong, but still with a bit of flex, and has the characteristic slipperiness of HDPE. Plastic bags are made out of the same plastic, but it's a lot harder to reduce them into nice coherent chunks of plastic. It is doable, though.

Don't try to build a ball mount. I tried to improvise one out of some PVC couplings and a wood ball, like others have done, didn't work very well. Even if it had worked the result would have been extremely crude to use without knocking the mount out of alignment, and severely limited altitude-wise (can't aim high enough), Just pay the 17$ for a solid mount, it's worth it. You get a quick release plate, too.

Stellarium is the shit. Seriously. It's awesome being able to recognize some constellations under your own power. Also, it lets you see where exactly the NCP is relative to polaris - remember to invert the position if using a finder scope. Also helpful is this diagram from celestron, as used in their polar alignment scope. Note, that diagram represents the view through a telescope or finder scope.

I'd use particleboard/plywood if I were to do this over (still may do sometime in the future). There's some warping in the wood I used, and it's not perfectly consistant. Price was right, though...

It is a cast iron motherfucker to focus the EF-S 55-250mm STM finely. Seems to have an acceleration factor based on how fast you turn it.

Before pegging an exposure as poorly focused, try a short exposure with extremely high ISO sensitivity first. If the stars are sharp in that, you've got a mount problem, either alignment, tracking, or vibration; not a focus problem.

The price for the "polarie" type trackers is criminal; I intend to try to build one sometime as they aren't much more than a camera mount being rotated at a constant rate.

Links

Cloudbait Observatory Computerized tracker
A Smart "Barn Door" Drive for Astrophotography
Triangle Calculator
Stellarium, a planetarium on your computer!
DeepSkyStacker - exposure stacking
Curved Rod tracker & **build instructions**
Dave Trott's write up of his Double Arm design

2014 ZZJ

return to top